Algoritmos http://www.lfcia.org/alg/ 2007-2008

Ingeniería Informática
Ingeniería Técnica de Informática de Gestión

Práctica 4

Fecha límite de entrega: viernes, 30 de noviembre de 2007

Ordenación por montículos

Algoritmo de *Ordenación por Montículos* usando un *montículo de mínimos*:

```
procedimiento Ordenación por montículos (V[1..n])
  crear_montículo (V, M);
  para i := 1 hasta n hacer
    V[i] := ObtenerMenorValor(M);
    EliminarMenorValor(M)
  fin para
fin procedimiento
```

donde el procedimiento para *crear un montículo* a partir de un vector cualquiera es el siguiente:

```
procedimiento crear_montículo (V[1..n], M)
{Entradas: el vector V[1..n] con cuyos datos se construirá el montículo
                    y el montículo M (estructura de datos abstracta)}
{Salida: el montículo M}
  Copiar V[1..n] en M[1..n];
  para i := n div 2 hasta 1 paso -1 hacer
    hundir(M,i)
  fin para
fin procedimiento
```

1. Implemente la estructura de datos *montículo* utilizando una instancia de la clase `java.util.Vector` para almacenar los elementos (figuras 1 y 2), y valide el correcto funcionamiento de cada uno de los métodos.

```
public interface IMonticulo {
    public void crear(Comparable [] v);
    public Comparable obtenerMenor();
    public void eliminarMenor();
}
```

Figura 1: La interfaz `IMonticulo`

2. Implemente el algoritmo de *ordenación por montículos* (figura 3), y valide su correcto funcionamiento.

3. Ejecute el algoritmo de ordenación por montículos con vectores de distintos tamaños n (500, 1000, 2000, 4000, 8000 ...) y con tres diferentes situaciones iniciales: (a) el vector ya está ordenado en orden ascendente, (b) el vector ya está ordenado en orden descendente, y (c) el vector está inicialmente desordenado.

```
public class Monticulo implements IMonticulo {
    private java.util.Vector datos = new java.util.Vector();
/*
...
*/
    public Comparable obtenerMenor(){
        if (datos.size() > 0)
            return (Comparable)datos.firstElement();
        else
            return null;
    }
/*
...
*/
}
```

Figura 2: La clase `Monticulo`

```
public class OrdenacionMonticulos implements Ordenacion {
/*
...
*/
}
```

Figura 3: La clase `OrdenacionMonticulos`

4. Calcule empíricamente la complejidad del algoritmo para cada una de las diferentes situaciones iniciales del vector.